Machine Learning Exploration: Decision Trees

Huang Weiran

Beijing Univ. of Posts and Telecomunications *

Last Update: July 24, 2019

本文很大程度上参考了《统计学习方法》《机器学习》等书籍。

1 What is a Decision Tree?

决策树（Decsion Tree）是一类常见的机器学习方法，它是用来对实例进行分类的一种树形结构。在这棵树上，每一个内部节点（internal node）

*This work is done while visiting Penn State University.

表示一个特征或属性，而叶节点（leaf node）表示一个类。下图就是一个决策树的例子。如《统计学习方法》书中提到的，决策树的根节点到叶节点的每一条路径对应着一条if- then规则，这一点看了下面的图应该是很显然的。

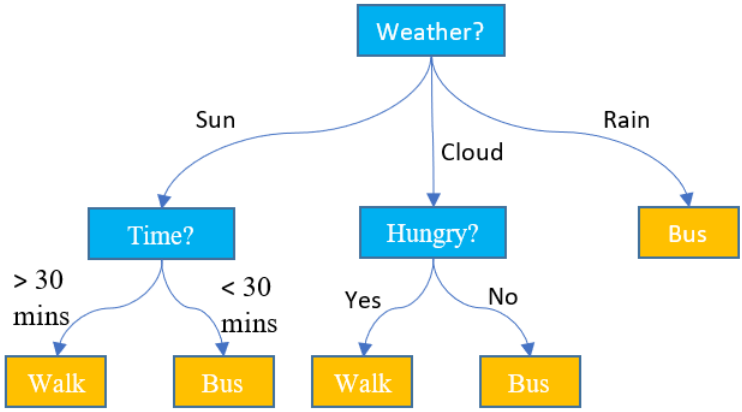

Figure 1: Decision Tree Example

从另一个角度看，决策树做的其实是对空间的划分。这一点可以很清楚地从Figure 2 上看出来，不再做更详细的解释。

在实际学习中，我们会有一组数据集

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

其中每一个x_i都是一个特征向量，y_i则是这个实例对应的类标签。这个特征向量通常不会是深度学习里稠密的embedding，而更类似词袋模型（bag of words）中的特征。依旧以上图为例，一个特征向量有三维，分别代

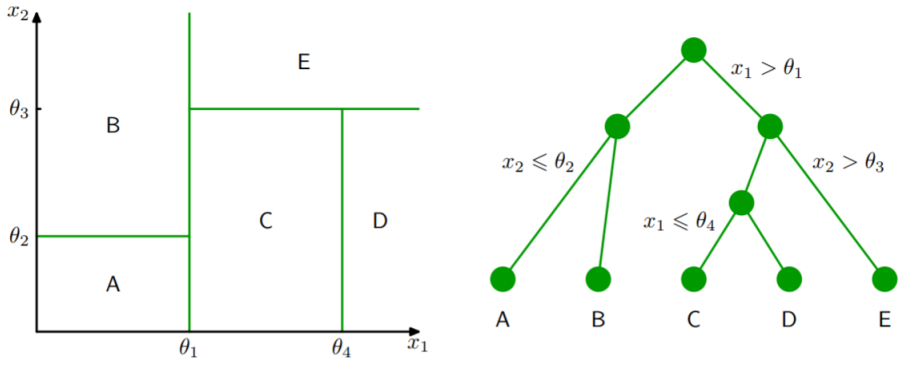

Figure 2: Decision Tree is splitting area

表Weather，Time和Hungry；这里假定分别用 1，2，3代表第一维（即Weather）上的Sun，Cloud，Rain，1和0分别代表第三维（即Hungry）上的Yes和No。那么如果我们现在有一个特征向量是[2, 18, 0]，那根据上面给出的这棵决策树，它将会被分类为Bus。有了这个例子，我们实际上已经知道了在已有一棵决策树的前提下，如何进行分类。

2 How to Build a Decision Tree?

在决策树的学习中，我们需要做的是从训练集中归纳出一组规则来生成决策树。通常与训练集不矛盾的决策树可能有多个，我们需要找到一个与训练集矛盾较小同时又有很好泛化能力（即对于其他数据的分类依然奏效）的决策树。决策树的构造通常是自上而下递归完成的，这里简单描述一下其步骤。

我们先考察递归过程什么时候会返回。

(1). **当前节点下所有数据的类别已经相同。**这时候不需要再分了。
(2). **当前已经没有剩余可用来划分的属性。**如果所有属性都已经用于划分，尽管没能达到情况1的结果，划分也不得不停止。这时候的做法是把该节点的类别定为其对应数据子集中样本最多的类。
(3). **当前节点无对应数据。**

接着这里对生成过程做一个非形式化的描述。

我们把所有训练数据都放在根节点，在根节点处选择一个最优特征，根据这个特征将训练集分为几个子集。如上例，我们首先根据Weather属性上的取值，将训练集分为了三个子集。考察上面的递归返回条件，如果满足，则对应的节点为叶节点；不满足则继续选择当前的最优特征，重复上述步骤。最后每一个子集都被分到叶节点上，这就生成了一棵决策树。

而怎样选择上面说到的“最优特征”则是最关键的部分。一般来说，我们会希望一个节点上样本的纯度（purity）越高越好，也就是说希望某一节点上尽量是同一类样本。基于这个目标，我们通常有三个标准：

- 信息增益（information gain）
- 增益率（gain ratio）
- 基尼指数（Gini index）

这三种分别对应了三种不同的决策树生成算法：ID3，ID4.5，CART。

2.1 Information Gain

在信息论中，熵（entropy）是刻画随机变量不确定性的度量，对随机

变量X，其熵的定义为：

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

这里我们定义$0 \log 0 = 0$。则可以看出，当X只有两个取值时，对其中的一个取值，如果p等于0或1，其熵都为0，表示随机变量完全无不确定性；而p等于0.5时，熵最大，表示其不确定性最大。**因此，我们注意到，熵很适合用来度量集合纯度。**

另一个概念是条件熵（conditional entropy）。条件熵$H(Y|X)$表示的是在已知随机变量X的情况下 Y的不确定性：

$$H(Y|X) = \sum_{i=1}^{n} p_i H(Y|X = x_i)$$

与刚才一样，我们注意到，条件熵很适合用来刻画在某特征给定的条件下对数据集D进行分类的不确定性。而$H(Y) - H(Y|X)$刻画的则是特征A对减少分类中不确定性的贡献，这个差被称作互信息（mutual information）。在决策树学习中，所谓信息增益就等价于类和特征的互信息（即Y是类，X是特征）。

ID3算法就是在每次递归时，选择信息增益最大的特征A_g作为节点的。（在《统计学习方法》中稍有不同，这个A_g 的信息增益必须达到某一个阈值；否则直接停止分类。后面C4.5和CART也是同样的道理，之后不再赘述。）

2.2 Gain Ratio

增益率是对信息增益的一个改进。因为以信息增益为基准会偏向于选择取值较多的特征，用一个例子来说明这样存在的问题：假如我们增加一

个特征是每个样本的标号，每个样本的标号都是不同的，如果以信息增益为标准，我们的算法会选择标号来进行分类，这样最后每个叶子节点中仅有一个样本。

可以用信息增益除以某个能表示特征取值多少的值来缓解这个倾向。这个值我们可以采用数据集D对于某特征A的取值的熵，即

$$IV(A) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

不过增益率又对取值较少的属性有偏好。因此C4.5采用的划分原则是：先找出信息增益高于平均值的属性，再从中选择增益率最高的。（这里《机器学习》和《统计学习方法》中描述不一致：后者仅以增益率为标准；个人感觉前者说的这种启发式方法更合理些。）

2.3 Gini Index

CART决策树（Classification and Regression Tree）既可以用来做回归也可以用来做分类。在它用来做分类的时候，使用基尼指数来选择划分属性。基尼指数反映的是从数据集D中随机抽取两个样本，其标记不同的概率：

$$Gini(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

在特征A的条件下，数据集D的基尼指数定义为：

$$Gini(D, A) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v)$$

不过由于CART决策树是二叉树，V实际上就等于2。它反映的是经过属性A划分后的纯度，数字越小纯度越高。

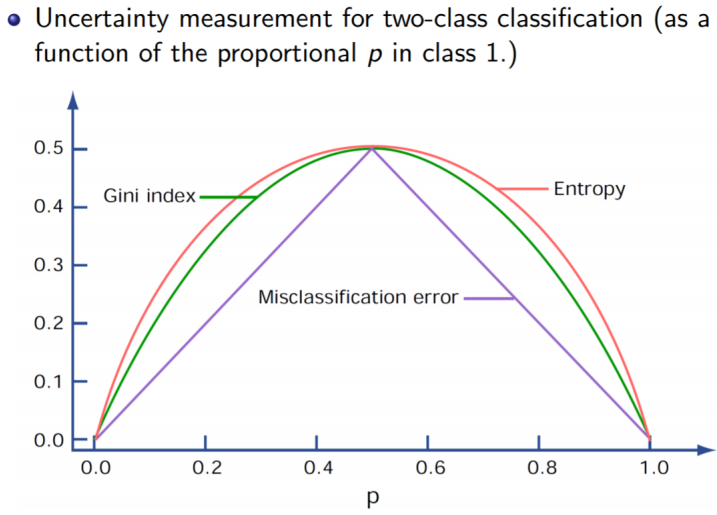

Figure 3: Comparison: Gini, entropy and error rate

Figure 3是在二分类问题上几种指标的比较，值得一提的是这里为方便比较取了熵的$\frac{1}{2}$。可以看到基尼指数和熵的效果差不多，但我们知道基尼指数计算要更容易些。当然这二者还是存在一些差异的，只是这里没有必要提这种小众场景。

不妨也关注一下CART回归树。我们之前提到过，决策树做的其实是空间的划分，在回归问题上也是如此。假设已经将输入控件划分为M个单元$R_1, R_2, \ldots, R_M$，并且每个单元有一个输出值c_m。简单来说，如果一个样本被划分到R_i上，它的输出就会是c_i。

当这个划分固定时，可以用MSE反映回归的预测误差，这样容易知道每个单元上的最佳输出值是该单元上所有样本取值的平均数。

由于CART决策树是二叉树，每次划分只需要一个变量（即x中的一维）即可产生两个子空间。至于究竟选哪个变量，切分点取多大，则需

要逐一扫描，使下式满足：

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_1}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

中括号内的两个最小值我们已经知道是该子空间上输出的平均值；而外部的切分变量和切分点的选择，则需要逐一扫描。递归地对子空间重复上面的步骤，直到划分完成即可（比如子空间数达到约定）。过程示意图如Figure 4所示。

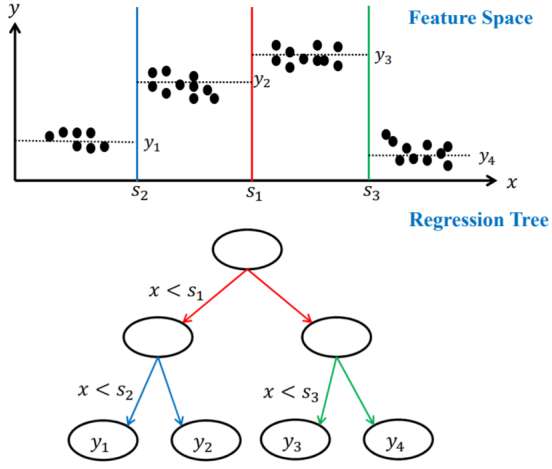

Figure 4: Regression Tree and its Feature Space

2.4 Pruning of a Decision Tree

当决策树分支过多时，可能会出现过拟合现象。我们可以通过主动剪去一些分支来降低过拟合风险，这种操作就叫做“剪枝”。剪枝策略包括预剪枝和后剪枝。

预剪枝实际上是一种贪心算法。它在每一次划分前先判断当前划分能不能带来泛化性能的提升（以验证集精度为反映），如果不能则停止划分，并标记当前节点为叶节点。但这种贪心策略可能导致某些暂时无法带来提升、后续操作可能有效的分支没法展开，模型因此可能欠拟合。

后剪枝则是在决策树生成之后，对内部节点进行考察。如果将该节点对应的子树替换成叶子节点能提升泛化性能，则替换。《统计学习方法》中只提到了后剪枝，并且方法并非通过验证集，而是通过设定损失函数（对于ID3和ID4.5）。对CART的剪枝有所不同，但我懒得看了。一个实际的原因：连Scikit Learn里都没有后剪枝，那还是别花时间看这个了吧。